AL/HR-TP-1995-0040

DESKTOP DECISION TRAINING (DDT) SYSTEM DESIGN DOCUMENT

Brian Van de Wetering

Systems Engineering Associates
2204 Garnet Avenue Suite 303
San Diego CA 92109-3771

Sharon K. Garcia

HUMAN RESOURCES DIRECTORATE
TECHNICAL TRAINING RESEARCH DIVISION
7909 Lindbergh Drive
Brooks AFB TX 78235-5352

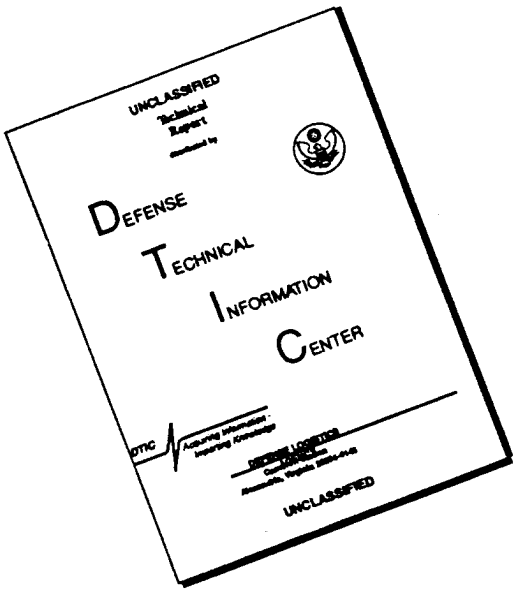
January 1996

Interim Technical Paper for Period January 1994 - December 1994

AIR FORCE MATERIEL COMMAND
BROOKS AIR FORCE BASE, TEXAS

ARMSTRONG LABORATORY

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

SHARON K. GARCIA
Project Scientist
Technical Training Research Division

R. BRUCE GOULD
Technical Director
Technical Training Research Division

JAMES B. BUSHMAN, Lt Col, USAF
Chief, Technical Training Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1996	Interim - January 1994-December 1994

4. TITLE AND SUBTITLE

Desktop Decision Training (DDT) System Design Document

5. FUNDING NUMBERS

C - F33615-91-C-0007
PE - 62205F
PR - 1121
TA - 10
WU - 76

6. AUTHOR(S)

Brian Van de Wetering
Sharon K. Garcia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Systems Engineering Associates
2204 Garnet Avenue, Suite 303
San Diego, CA 92109-3771

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Armstrong Laboratory
Human Resources Directorate
Technical Training Research Division
7909 Lindbergh Drive
Brooks Air Force Base, Texas 78235-5352

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL/HR-TP-1995-0040

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Dr. Sharon K. Garcia; (210) 536-2932

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper describes the software design of the Desktop Decision Training (DDT) system. It documents how the system's software implements an instructional strategy to train complex decision-making skills to personnel in the Logistics Command and Control domain. This paper is intended to be a pragmatic guide for the DDT's software programmers and technical management personnel.

14. SUBJECT TERMS

Command and Control
Decision Making
Instructional Design
Instructional Strategy
Logistics
Simulation

15. NUMBER OF PAGES
58

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Table of Contents

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

PREFACE

This paper describes the software design of the Desktop Decision Trainer (DDT) developed in support of "Desktop Training for Logistics Command and Control (LC^2)," research and development effort. This project is being accomplished under Contract No. F33615-91-C-0007, with Systems Engineering Associates (SEA), San Diego, CA. Management of this project is being provided by the Human Resources Directorate, Technical Training Research Division, Instructional Design Branch (AL/HRTC).

Summary

Logistics Command and Control (LC^2) units must ensure that core and augmentee personnel are fully trained in the critical combat skills of decision making. At present, existing training capabilities are inadequate. They consist primarily of expensive and manpower-intensive exercises, which afford only sporadic training opportunities. These opportunities are considered insufficient to achieve and maintain the skill levels required for successful combat operations. The need for more accessible, more affordable, and less manpower-intensive training continues to exist.

In 1991, the USAF Logistics Plans and Concepts Directorate (USAF/LGXX) tasked the Human Resources Directorate (HR) to develop an improved training technology for Logistics Command and Control Centers throughout the United States Air Force. The objective was to provide a means of training logistics personnel in the combat-critical task of decision making. In response, HR let a contract with Systems Engineering Associates (SEA) to produce a desktop decision trainer which would provide individual instruction and enable students to practice solving realistic logistics problems within a simulation environment. The project began in February 1992, and will conclude in February 1997.

This paper describes the software design of the decision trainer, and in particular, how the system implements the instructional strategies developed to teach decision-making skills.

1. Purpose and Scope

This document describes the software design of the Desktop Decision Trainer (DDT). The document is a companion to the DDT System Requirements Document (Brecke & Garcia, 1995). The System Requirements Document identified system features required to implement the DDT's instructional methodology. This document describes how the system's software implements those features. It is intended as a pragmatic guide for the DDT's software programmers and technical management personnel.

Section 3 presents an overview of the DDT's software architecture. It describes each of the system's primary processes and databases and their interactions. Sections 4 and 5 provide more detailed descriptions of the software partitions implemented in Beta Version 0.1 of the DDT.[1] These partitions include the system's lessons and exercises. Section 4 describes the system's exercise software. This software was produced using object-oriented design techniques. Section 4 identifies the major objects used to implement DDT exercises and describes each object's purpose, attributes, and operations. Section 5 describes the system's lessons. This software was produced using ToolBook (Ver. 1.53), a commercial authoring package produced by Asymentrix, Inc. Section 5 describes the organization of DDT lessons in relation to ToolBook's book- and page-oriented authoring paradigm.

Sections 1, 2, and 3 are intended to be reviewed by all readers of this document. Readers of Section 4 should be familiar with object-oriented software design and implementation. Readers of Section 5 should be familiar with the operation of ToolBook. Each section includes references that provide this background information.

---

[1]Beta Version 0.1 of the DDT was released on 31 December 1993. It is a preliminary prototype prepared during Phase 2 of Contract F33615-91-C-0007 and includes two lessons and three exercises.

2. Architecture Overview

A software system like that of the DDT is a collection of software components that work together to achieve common goals. The DDT's System Requirements Document (Brecke & Garcia, 1995) describes the DDT's goals. This section provides an overview of the DDT's organization, or architecture, by identifying each DDT software component and its relationship with other components.

The DDT's architecture has been fashioned to:

- Satisfy requirements stated in the DDT's System Requirements Document.
- Facilitate implementation and maintenance of the system's software.
- Separate features supporting authoring of instructional strategy from features supporting authoring of lesson content.
- Maximize the use of commercial off-the-shelf (COTS) software.
- Minimize the amount of custom software that must be developed.
- Minimize potential disruption to the system caused by the inevitable evolution of computer operating systems and hardware.
- Ensure adequate performance on a personal computer (PC) platform that will be common on the desktops of Air Force logistics personnel during the summer of 1994.
- Allow emerging multimedia effects to be incorporated into the system with minimum disruption.

The DDT presents lessons and exercises to a student. Lessons present knowledge, while exercises provide a simulated logistics problem-solving environment to apply this knowledge. Figure 1 illustrates this organization of instruction.

Figure 1. The DDT's instruction.

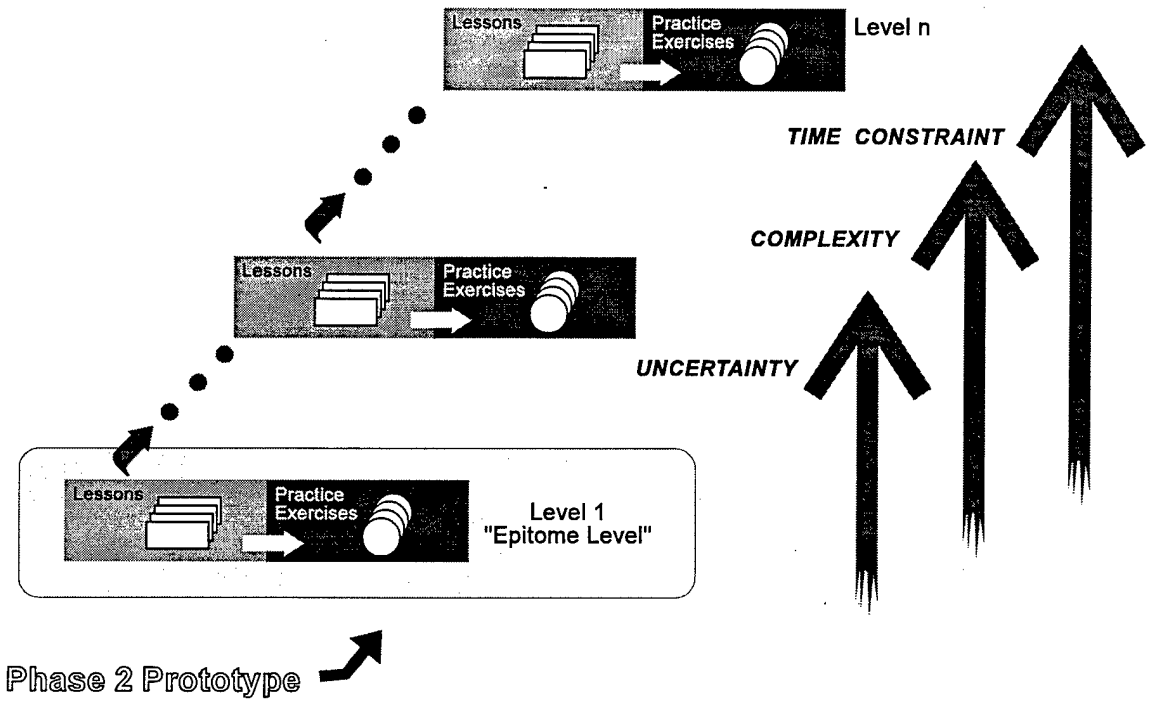
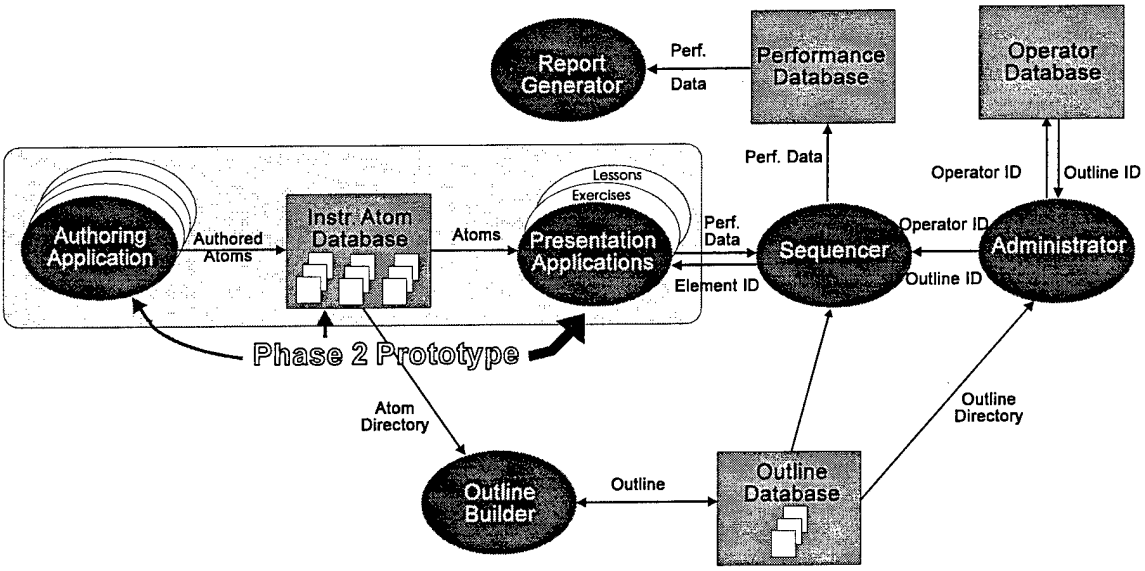


Figure 2 illustrates the software architecture used to implement lessons and exercises. "Ovals" represent processes and programs in this diagram, and "boxes" represent databases.

The system's instructional atom database includes all of its instructional content. This database is made up of instructional atoms, which are individual lessons and exercises (Figure 2) Instructional atoms are created by authoring applications and are displayed by presentation applications. This architecture introduces minimal constraints upon the content of lessons and exercises (instructional atoms); it allows virtually any kind of instruction to be used, as long as the instruction is accompanied by an authoring program and a presentation program. These programs may be COTS products or custom software. Beta version 0.1 of the DDT, for example, includes a COTS product,

ToolBook, to author and produce lessons, and a custom software program to present exercises.

Figure 2. The DDT databases and software processes.



The outline database illustrated in Figure 2 includes instructional strategies for courses. Each strategy identifies the instructional atoms that will be presented in a course and constrains their presentation. For example, an instructional strategy that includes minimum learner control may dictate a strict presentation sequence for a course's atoms, while a strategy that provides maximum learner control may remove all constraints upon the sequence of atom presentation. The DDT's instructional atom database and its course outline database separate the system's instructional content from instructional strategy.

Course outlines are produced by an outline builder and are executed by the system's sequencer. The outline builder resembles an authoring application, but instead of enabling the production of instructional atoms, it enables production of course

outlines. The DDT's sequencer is responsible for invoking presentation applications (and thus displaying instructional atoms) in accordance with a course outline.

Information regarding a learner's performance is recorded by the sequencer in the DDT's performance database. The system's report generator organizes performance data for display and printing.

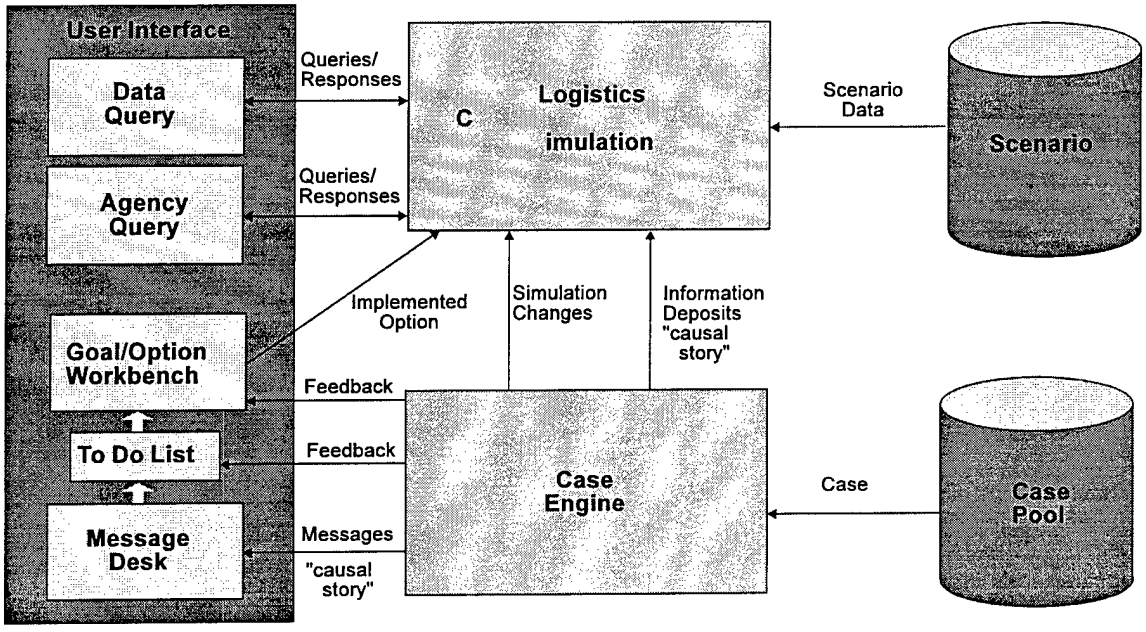
The DDT's final database, the operator database, identifies operators and classifies them as learners, instructors, authors, or researchers. It also associates each operator with a course and a strategy. The system's administrator process manages this database and provides the system's login services.

Beta version 0.1 of the DDT includes implementations of the system's instructional atom database, as well as the authoring and presentation applications needed to present the system's initial lessons and exercises. The following sections discuss the organization of these components in more detail. Section 4 describes the custom software that presents DDT exercises, and Section 5 describes the use of COTS software to present initial DDT lessons.

3. Exercise Architecture

Figure 3, depicts the DDT's exercise presentation software. It consists of five elements: a logistics simulation, a case engine, a user interface, a scenario database, and a case pool. The logistics simulation provides a low-fidelity simulation of a logistics environment. The DDT's case engine provides a mechanism for introducing instructor-specified problems into the simulation. The user interface provides a "point-and-click" environment that allows a student to work on simulated logistics problems. The scenario in Figure 3 is a large data file that contains all of the simulation's initial conditions. The case pool is a data file containing problem specifications for the case engine. Each of these software partitions and data files is described in more detail in the following sections.

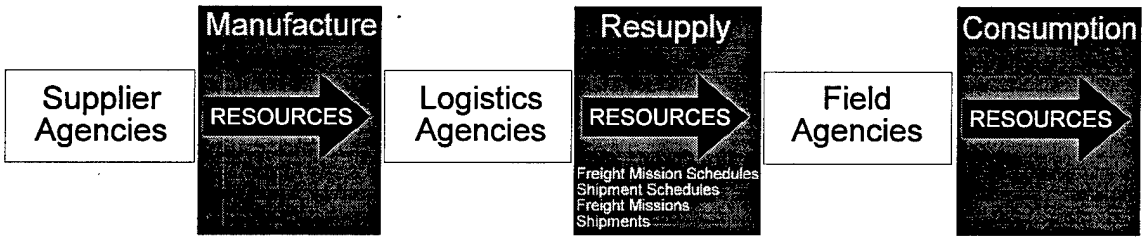
Figure 3. DDT exercises.



3.1. Logistics Simulation

The DDT's logistics simulation provides the student with a set of simulated objects that correspond to items in an actual logistics environment. The student interacts with these objects through the user interface. In Figure 4, these objects include resources, such as fuel or transport planes, agencies, shipments, freight missions, and schedules. The DDT's simulation is designed to emulate a logistics operation's sustainment phase, in which manufacturers provide resources to logistics agencies, who in turn resupply operational agencies in the field, who account for the consumption of resources. The simulation software has been designed and implemented using object-oriented programming techniques. Its software partitions therefore consist of objects, associations between objects, and events that occur as these objects interact. Section 3.1.1 describes the major software objects used in the simulation. Section 3.1.2 describes the primary associations or relationships between those objects. And Section 3.1.3 describes events that occur as objects interact. Appendix A provides more detailed descriptions of these objects, associations, and events.

Figure 4. DDT's logistic simulation.



3.1.1. Objects

Objects are software modules that include both data and program logic. They are used in the DDT's logistics simulation to represent items that are familiar to logisticians, such as operational units, supplies, and transportation vehicles. Objects are a useful way to organize a software in a simulation like the DDT's logistics simulation because they allow programmers to construct complex software behaviors using relatively small,

autonomous modules. Objects are also useful because they allow programmers to organize software in a way that relates directly to the actual situations the software is supposed to model. The primary objects used by beta version 0.1 of the DDT are described below. Appendix A contains more detailed descriptions of these objects.

Agency

An agency is a place or an organization. It may be an operational unit such as a starfighter wing, or it may be a supply depot, a transportation command, or a supplier/manufacturer. An agency can be an aggregate of several other agencies. In this case, the agencies are considered to be collocated and there is no need to transport resources between them. Agencies own or have jurisdiction over resources and other agencies.

Resource

A resource is any physical entity, e.g., starfighters, fuel canisters, beer, compressed biomass cakes, fighter pilots, or mechanics.

Vehicle

A vehicle is a resource capable of moving commodities from one agency to another.

Commodity

A commodity is a resource whose supply the student must manage.

Consumable

A consumable is a commodity that is used up, e.g., beer, fuel, and food. Consumables have a one-way transportation flow from supplier to consumer.

Repairable

A repairable is a commodity that is refurbished after it has been used, e.g., starfighter engines, guidance systems, and R2D2 units. Repairables have a two-way

transportation flow between the supplier and the consumer; refurbished units are sent to the consumer and spent units are returned to the supplier.

Proto-Resource

A proto-resource is an object that is a pattern for an actual resource. There is a proto-resource for each type of resource. Examples of resource types are fuel canisters, starfighters, hyper-warp freighters, and mechanics. A proto-resource contains the unchanging attributes that are common to all resources of a particular type. A proto--resource is associated with the indefinite article "a," e.g., "I need a starfighter engine." Resources, on the other hand, are associated with the definite articles "the" and "that," e.g., "I am shipping that starfighter engine." The subclass tree for proto-resources is homomorphic with the subclass tree for resources.

Route

A route consists of two agencies and a list of one or more proto-vehicles and describes a transportation corridor between the agencies that supports the listed proto-vehicles.

Freight Mission

A freight mission consists an origin agency, a destination agency, a vehicle, and an ordered list of routes. A freight mission describes a trip from the origin agency to the destination agency following the routes on the ordered list.

Freight Mission Schedule

A freight mission schedule consists of an origin agency, a destination agency, a proto-vehicle, and an ordered list of routes. A freight mission schedule may also be connected to zero or more shipment schedules. A freight mission schedule describes a recurring freight mission and originates freight missions according to its schedule.

Shipment

A shipment consists of an origin agency, a destination agency, a commodity, one or more freight missions, and an ordered list of zero or more transfer point agencies. A

shipment describes the movement of a commodity from the origin agency to the destination agency. A shipment must adhere to certain constraints of space and time. A commodity cannot be in two places at once, so all freight mission departure/arrival intervals at transfer points must be mutually exclusive. Since a commodity cannot magically jump from one place to another, adjacent freight missions in the ordered list must intersect at a transfer point. A shipment transfers ownership of the commodity from the origin agency to the destination agency.

Shipment Schedule

A shipment schedule is a pattern for shipments. A shipment schedule describes the recurring movement of a commodity from one agency to another; and it consists of an origin agency, a destination agency, a proto-commodity, one or more freight mission schedules, and an ordered list of transfer point agencies. The same time and space restrictions that apply to the shipment also apply to the shipment schedule.

Consumption

A consumption object consists of an agency and a proto-commodity and describes the consumption of that type of commodity by the agency. A consumption association has attributes of consumption rate and demand threshold. A consumption association object is responsible for destroying commodity objects and drains commodities out of the simulation.

Supply

A supply object consists of two agencies and a proto-commodity. One of the agencies is the supplier and the other is the recipient. The supply association describes the recurring flow of commodity objects from the supplier to the recipient. This transfer of commodities occurs by "magic," without the need for vehicles, freight missions, or shipments. The supply association is responsible for creating the commodity objects, and it pumps commodities into the simulation.

3.1.2. Associations

Associations are relationships between objects. For example, when a person is employed by a firm, the person and the firm enjoy an employment association. Associations are useful because they provide a convenient way to represent such relationships. Associations can be implemented as attributes of an object or as separate objects. For example, if a program has objects that represent people, these objects may have an attribute that identifies the firm that currently employs each person. Or the program may have separate objects, called employment objects, that identify each firm and person currently engaged in an employment relationship.

In the DDT, associations are implemented both as attributes and as separate objects, to provide greater flexibility and a more accurate simulation. In general, cases where the association itself has attributes are modeled as objects, for example a consumption association between an agency and a commodity. Associations modeled as objects have already been discussed in section 3.1.1. The remaining associations modeled in the DDT as attributes are summarized below.

Location

Location is an association between an agency and a resource, or between a freight mission and a resource. Location associations are changed in response to arrival and departure events. Aggregation of agencies implies collocation. For example, if Starbase 10 consists of the 11th SSFS, the 21st SSFS, the 23rd IC2S, and the 24th SSS, then these agencies would be treated as if they were in the same place; resources located at Starbase 10 could be accessed by the 24th SSS without transportation.

Ownership

Ownership is an association between an agency and a resource, an agency and another agency, an agency and a freight mission, an agency and a freight mission schedule, an agency and a shipment, or an agency and a shipment schedule. In an ownership association between agencies, one is the owner and the other is the chattel. Ownership establishes jurisdiction; only an owner can grant permission for a resource.

Ownership is transitive. For example, if agency A owns agency B and agency B owns vehicle C, then agency A owns vehicle C.

Instantiation

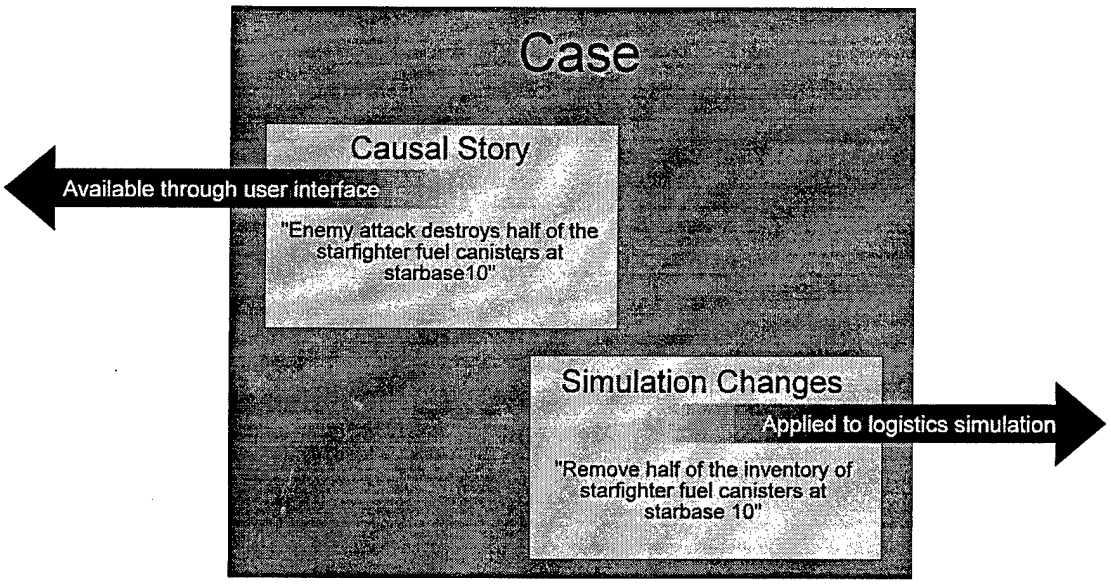
Instantiation is an association between an object and another object that serves as its pattern. A resource is always an instance of a proto-resource and inherits attributes from it. Likewise, a freight mission can be an instance of a freight mission schedule, and a shipment can be an instance of a shipment schedule.

3.2. The Case Engine

As illustrated in figure 5, a case consists of two main elements: the causal story, and logistics simulation changes. The story provides explanatory information about why certain logistics events (e.g., shortages, consumption increases, etc.) have occurred. In many cases, the details of the story influence how the student is expected to react to the problem. The elements that make up the story are structured to allow the student to "discover" the story and to permit evaluation of the student's discovery process.

Logistics simulation changes are a description of the changes that need to be made to the logistics simulation to "cause" the logistics event associated with case. Sections 3.2.1 and 3.2.2, respectively, discuss the causal story and the logistics simulation changes in more detail. Section 3.2.3 contains detailed descriptions of objects that make up a case.

Figure 5. DDT's case engine.

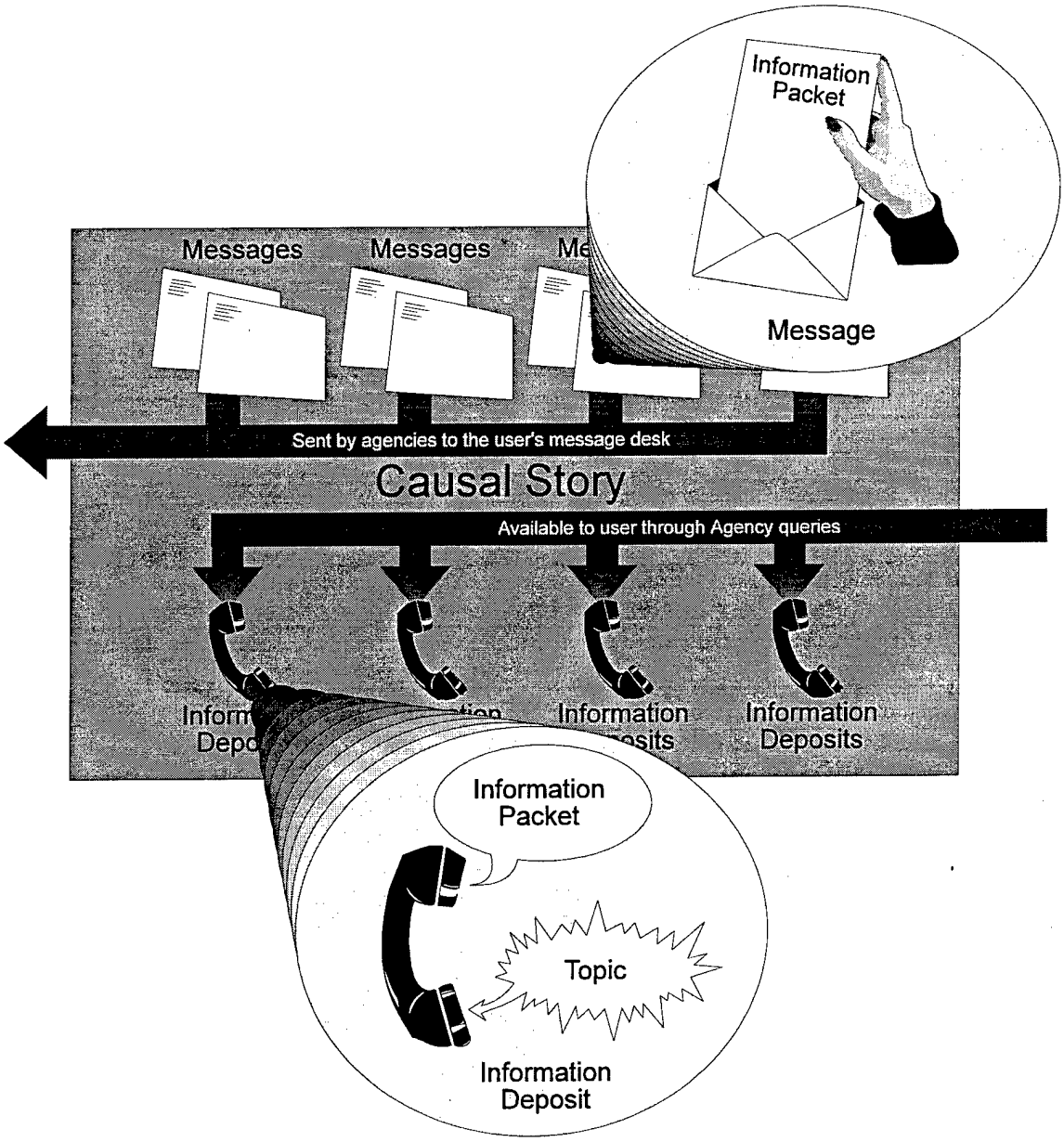


3.2.1. The Causal Story

As illustrated in figure 6, the causal story consists primarily of messages and information deposits. Messages, like mail, are unsolicited information. They arrive at the message desk during the course of a decision making episode (DME); the student can open them and view their contents and can file or otherwise dispose of them. Information deposits, such as those resulting from a student's phone call, are solicited communication. The student must explicitly query an agency to get the information contained in an information deposit.

Each information deposit is associated with a responding agency and an eliciting topic and contains information that is presented when the student asks the agency about that topic. A topic is a word or short phrase that can serve as the object of a student's query of an agency. This word or phrase will have appeared in some information previously viewed by the student.

Figure 6. A causal story.



Both messages and information deposits communicate their information using an information packet. An information packet is the smallest element of information and contains the actual text or pictures that present the information to the student. Currently, three types of information packets are envisioned: an object linking & embedding (OLE) packet, a query packet, and a text packet. An OLE packet will display OLE objects such

as Video for Windows, digital audio, or MIDI files. A text packet will display the contents of a simple text file. A query packet will display the results of a query of the state of some part of the logistics simulation. Only the text packet will be implemented in this phase.

3.2.2. Logistics Simulation Changes

Logistics simulation changes are a description of the changes that need to be made to the logistics simulation to "cause" the logistics event associated with a case. There are five different types of logistics simulations changes: adding resources, removing resources, tagging resources, and adjusting consumption and supply rates.

Resources can be added to the simulation. Adding resources might be necessary to give the student specific options for correcting a shortage. Resources can be removed from the simulation. If commodities are removed, it will cause a possible one-time shortage. If transportation resources are removed, it will cause a possible supply rate problem.

Resources can be tagged with information that relates both to the story and to the resources' behavior in the simulation. For example, food kits might be marked as contaminated and not available for consumption. This would cause a shortage of food kits. The food kits, however, would appear normal if the student formulated a query about food kits. Only when the student asked about an appropriate topic would the information about the contaminated food kits be revealed.

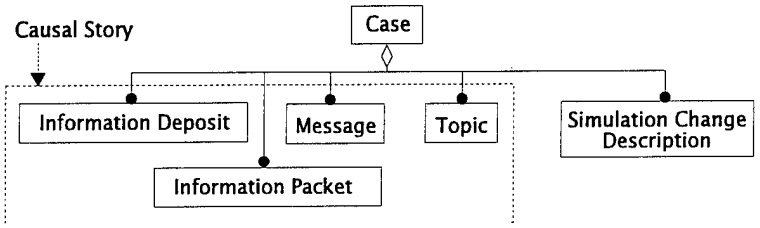
Consumption rates can be changed. An increase in consumption at a particular agency will cause a shortfall if the student does not adjust the appropriate shipment schedules. Supply rates can be changed. An decrease in supply from a particular agency will cause a shortfall if the student does not adjust other supply rates or adjust shipment schedules.

3.2.3. Case Engine Objects

Like the rest of the DDT's exercises, its case engine was designed and implemented using object-oriented programming techniques. Following are descriptions of the primary objects that make up the DDT's case engine. More detailed descriptions of the objects associated with a case appear in Appendix A.

Case Object

The following diagram shows the objects that make up a case. The case object is responsible for managing active topics and for providing feedback about problem identification and uncertainty reduction.



Message

A message has a subject, a delivery time that indicates when during the DME it should be delivered, and a sending agency. Some messages in a case are simply delivered when their delivery time arrives, while others are associated with information deposits and are delivered only after the student has discovered the deposit.

Information Deposit

An information deposit consists of several elements: an eliciting topic, an immediate response, a follow-up message, and a rebuff. The eliciting topic is the topic for which the information deposit is a response. The immediate response is sent to the

student as soon as the query is made. The follow-up message is sent some time after the initial query. The rebuff is presented if the user asks about the same topic again.

Topic

A topic is a word or a short phrase that can serve as the object of a user's query of an agency. The topic contains a description of the conditions under which it becomes available. For example, the topic of "destroyed fuel canisters" would become available after the student had viewed a message about an enemy attack at Starbase 10 or the student had read a report listing the shortage.

Information Packet

Information packets are the objects that contain the actual content of a message or information deposit. That information might be delivered as text, audio, or video. An information packet is tagged with information about the types of information it contains (e.g., situation, option set, option feasibility, option effects, distracter), allowing the case to provide feedback about the user's uncertainty reduction.

Simulation Change Objects

These objects describe and effect the changes to the logistics simulation necessary to "cause" the logistics event associated with a case. They contain the information necessary to describe the change.

3.3. User Interface

The purpose of the user interface is allow the user to interact with the objects that make up the logistics simulation and a case. The user interface has five primary components:

- The message desk, where incoming messages arrive and the user can read them.
- The data query screen, where the user can get information about the current states of commodities, freight missions, shipments, and schedules.

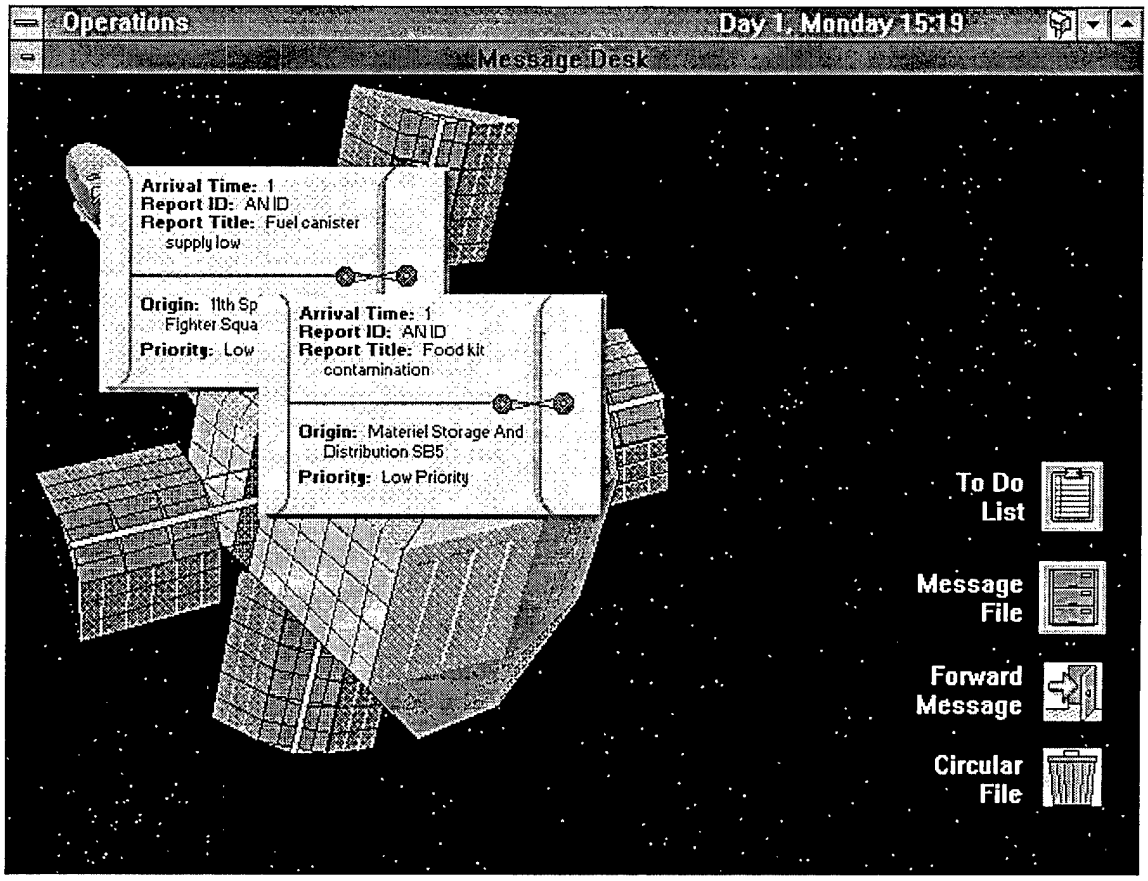
- The agency query screen, where the user can ask specific question of agencies.
- The "to do" list, where the user identifies and prioritizes the cases presented.
- The goal/option workbench, where the user sets goals and formulates solutions.

The user interface serves two functions: (1) It allows the user to navigate among these activities and, (2) Allows the user to perform these activities by interacting with simulation and case objects. The user interface runs under Microsoft Windows version 3.1 and is consistent with many user interface conventions in that environment. C code and standard Windows Application Program Interface(API) calls create the link between the simulation and case objects and what the user sees and does on the screen. A prototype created with ToolBook serves as the specification for the user interface. Sections 3.3.1 through 3.3.5 describe in more detail each of the user interface components.

3.3.1. Message Desk

The message desk provides an environment where the user interacts with message objects and the information packets they contain. These objects are described in section 3.2.3 and appendix A. The message desk is where the user reads messages and where messages first appear when they arrive. As illustrated in Figure 7, Windows API code represents each message object as either an open or a closed envelope on the message desk. The user can open and read a message by double clicking the envelope. When this happens, the API code that manages the envelopes asks the appropriate message to display its information packet.

Figure 7. Message desk screen.



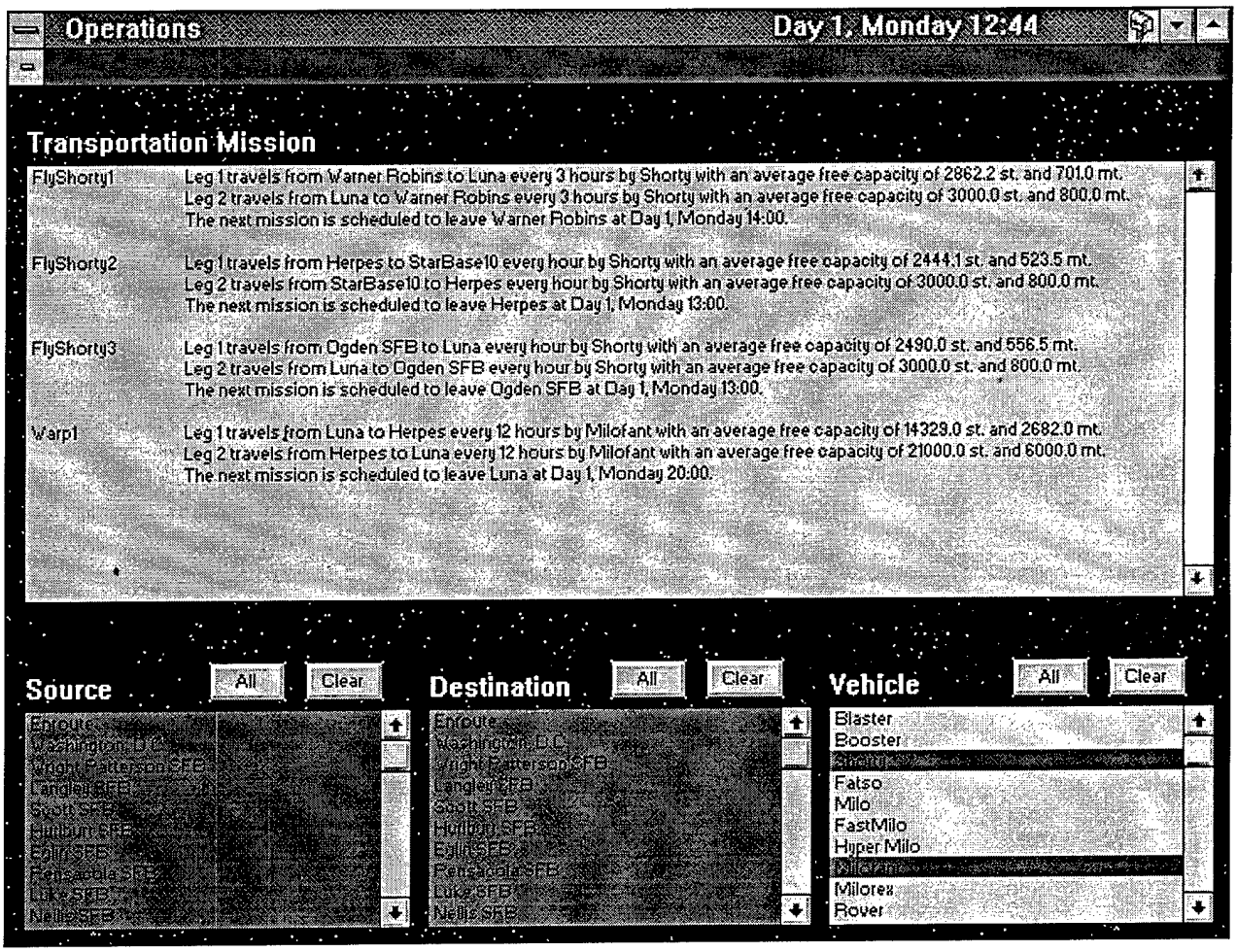
The user may also dispose of messages by dragging the envelope to one of the icons in the lower right corner of the screen. This causes the message object to be transferred to a different screen maintained by the user interface or to disappear entirely. Messages that are forwarded or put in the circular file disappear, while messages moved to the "to do" list or the message file appear subsequently on those screens.

The message desk is implemented as a maximized child window of the main screen. The message envelopes are child windows of the message desk with special callback functions that facilitate the drag-and-drop interaction. The image on the background of the message desk is a bitmap that is loaded at run-time.

3.3.2. Data Query Screen

The data query screens allow the user to find out information about objects in the logistics simulation. Four data query screens display information about commodities, vehicles, freight mission schedules, and shipment schedules. These simulation objects are described in section 3.1. Figure 8 shows the screen for freight mission schedules, but the other three look very much the same. The large list box at the top of the screen displays the results of the query, and the three small list boxes at the bottom allow the user to restrict the parameters of the query. In the freight mission schedule data query screen, the user can look at lists of freight missions based on their source agency, destination agency, or type of vehicle.

Figure 8. Data query screen snapshot.



In the example shown in figure 8, the query shows all freight mission schedules using the Shorty and Milofant transport ships. The user interface code fills the bottom three list boxes by looking at agency and proto-vehicle objects in the logistics simulation. Each time the user clicks on one of those three list boxes, the user interface looks for freight mission schedule objects in the logistics simulation that meet the specified criterion. The user interface then displays that information in the large list box. As with the message desk, the data query screens are maximized child windows of the main screen. The three boxes along the bottom of the screen are multiple-select list boxes, while the large viewing area is a single selection list box.

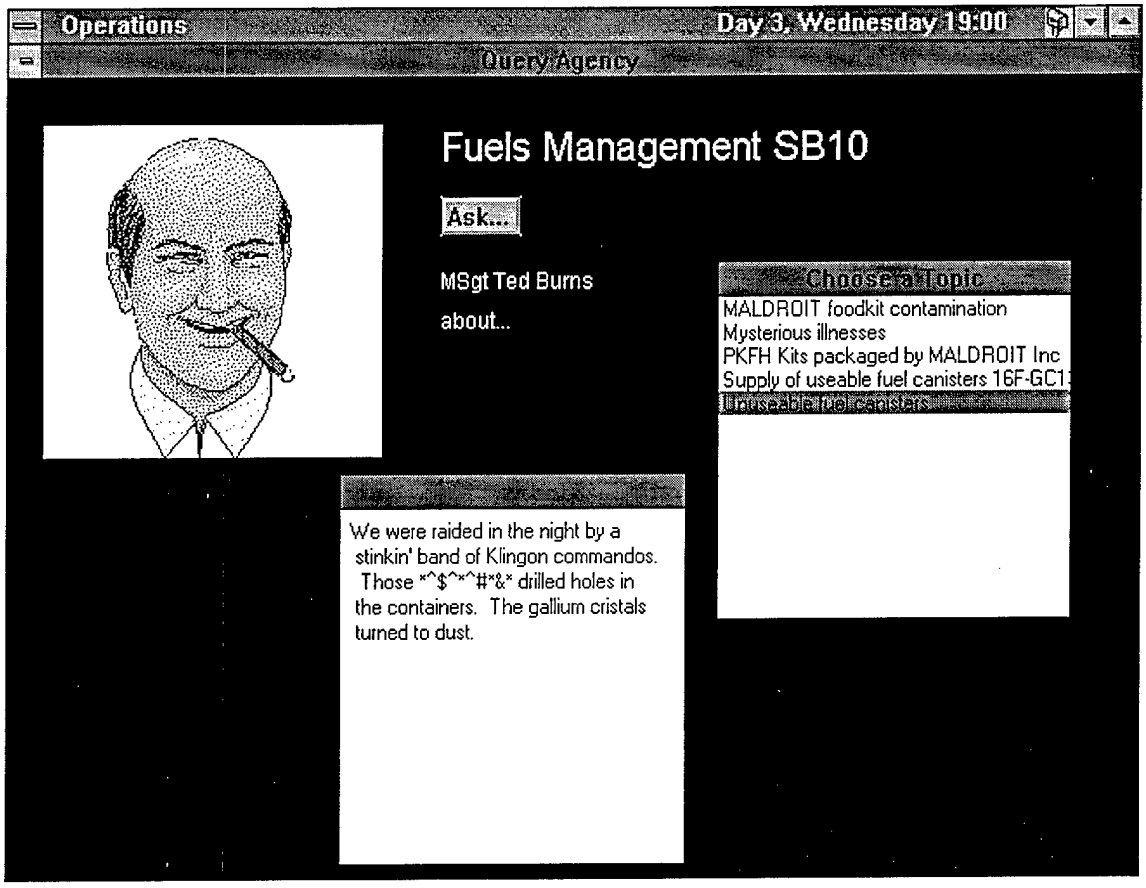
3.3.3. Agency Query Screen

The agency query screens are where the user interacts with information deposit and topic objects. These objects are described in Section 3.2.3. There is an agency query screen for each agency in the logistics simulation. They all, however, appear the same as the screen depicted in figure 9. The student accesses information deposits for the agency by choosing a topic from the topic list and clicking on the ask button.

If there is an information deposit at the agency for that topic, the immediate response will be displayed; otherwise, a default response will be displayed. User interface code looks at the case objects currently in the case engine to get a list of all the currently active topics and to determine if an appropriate information deposit exists.

The agency query screen is a maximized child window of the main screen. The box at the bottom of the screen that contains the response is a child window of the agency query screen that is created and managed by the information packet object.

Figure 9. Agency query screen snapshot.



3.3.4. "To Do" List

The "to do" list screen is where the user identifies and gives names to the cases that are presented during a DME. The case object is described in Section 3.2.3. The user should end up with a list of problems on the "to do" list that corresponds exactly to the cases presented in the DME and that are prioritized according to their urgency by examining the contents of each message and assigning it to the appropriate problem.

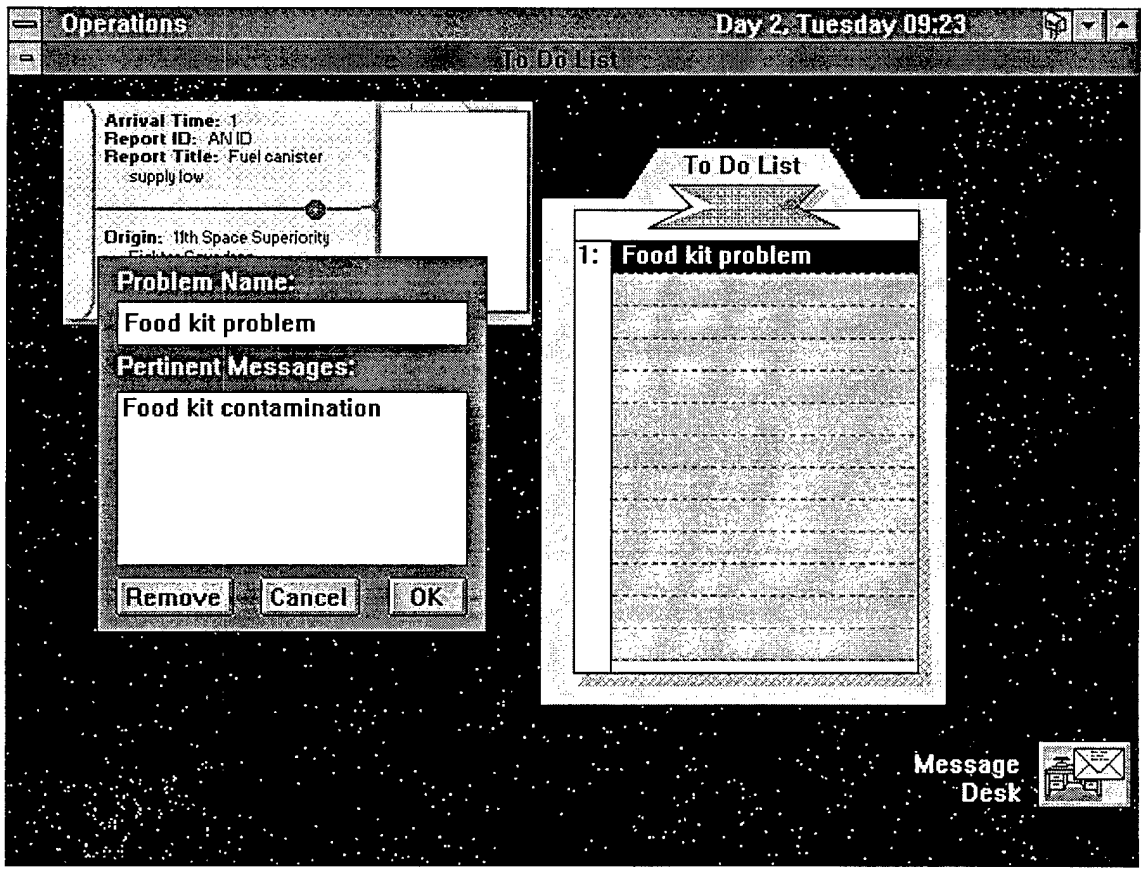
To accomplish this, the user may

- Create a problem by dragging an envelope onto the "to do" list and entering a new problem name into the resulting dialogue box.

- Double-click on a problem line on the "to do" list to bring up the dialogue box to change the name of the problem, or remove the problem by clicking on the remove button.
- Add pertinent messages to a problem by dragging the message into the lower part of the problem dialogue box.
- Rearrange (prioritize) the problems on the "to do" list by clicking and dragging a problem line to a new position.

The user may also double-click on messages and read them just as on the message desk. The problem dialogue box is implemented as a non-modal dialogue box. Each problem line on the "to do" list is implemented as a child window of the "to do" list screen with a callback function that facilitates the special drag-and-drop interaction of prioritizing problems.

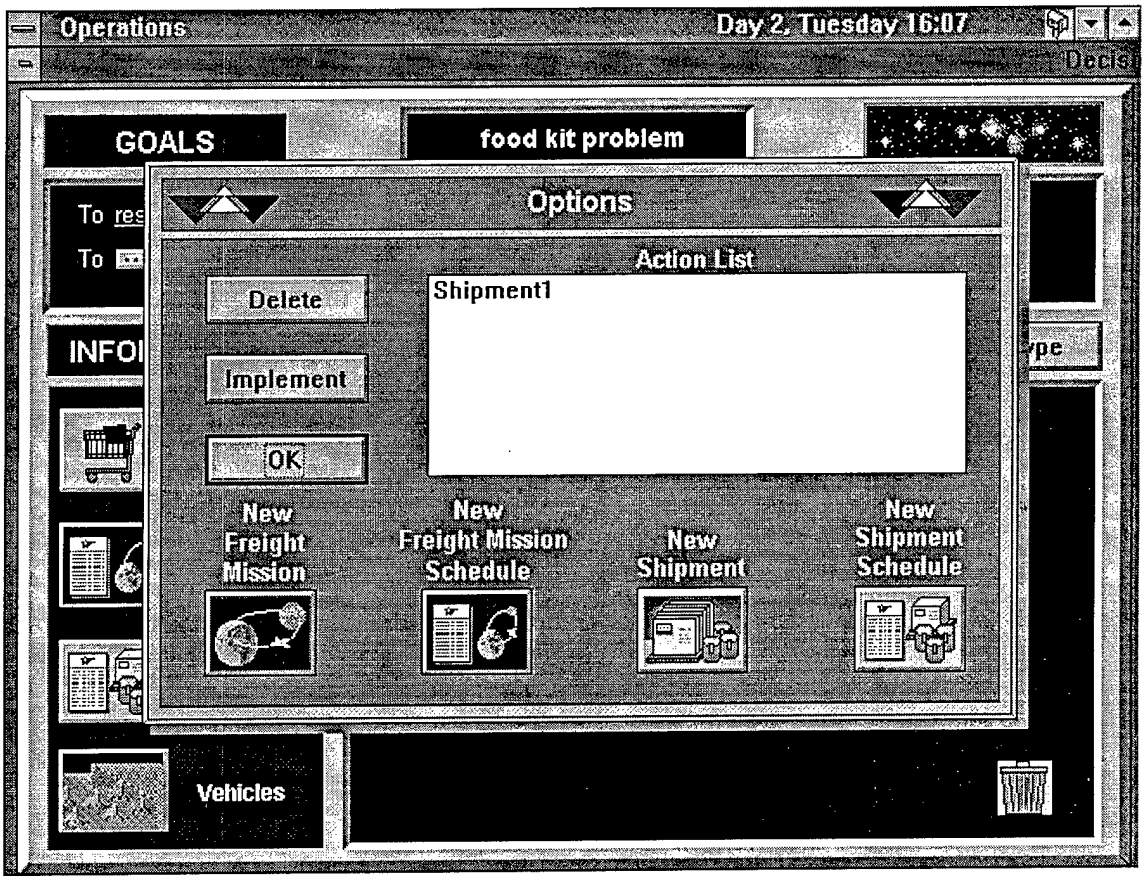
Figure 10. "To Do" list screen snapshot.



3.3.5. Goal/Option Workbench

The goal/option screen is where the user formulates solutions to the problem presented by the case object. To solve a problem, the user creates new logistics simulation objects. As illustrated by the screen depicted in figure 11, the user may create freight missions, freight mission schedules, shipments, and shipment schedules. These objects are described in section 3.1.1. A solution option will usually consist of a combination of several of these types of objects. For example, the user might need to create a new freight mission in order to have transportation for a new shipment.

Figure 11. Goal/option screen snapshot.



The user interface of the goal/option screen allows the user to specify all of the necessary attributes for these new objects. The attributes of new objects are stored temporarily until the user chooses to implement an option, then the objects are created

and introduced into the logistics simulation. The options box and the forms that allow the user to enter attributes for new objects are implemented as modal dialogue boxes.

3.4. Scenario Database

The scenario database is a large data file that contains specifications for all of the objects that make up the initial environment for the exercises (i.e., the scenario). This file contains specifications for instances of each type of object described in section 3.1.1. Proto-resources must be defined so that it is known what kinds of things exist in the world. The following shows definitions of a consumable and a vehicle.

```
$ CREATEFRAME FuelCakes INSTANCE
FuelCakes ASSERTRELATION AKO ProtoConsumable
FuelCakes ADDSLOTVALUE Name IS Fuel=Canister,=Compressed=Biomass
FuelCakes ADDSLOTVALUE Weight IS 2.0
FuelCakes ADDSLOTVALUE Volume IS 2.0
FuelCakes ADDSLOTVALUE Criticality IS 1
FuelCakes ADDSLOTVALUE Stockid IS =16F-CB54
FuelCakes ADDSLOTVALUE ColloquialName IS Fuel=Cakes
FuelCakes ADDSLOTVALUE ComType IS F_TYPE
FuelCakes ADDSLOTVALUE ShippingContainer IS PL
FuelCakes ADDSLOTVALUE QuanityPerContainer IS 100
FuelCakes ADDSLOTVALUE ShelfLife IS 30
```

$ CREATEFRAME FastMilo INSTANCE

FastMilo ASSERTRELATION AKO ProtoVehicle

FastMilo ADDSLOTVALUE Name IS FastMilo

FastMilo ADDSLOTVALUE Abbreviation IS LHLW

FastMilo ADDSLOTVALUE Drive IS WARP

FastMilo ADDSLOTVALUE WeightCapacity IS 5500

FastMilo ADDSLOTVALUE VolumeCapacity IS 1800

FastMilo ADDSLOTVALUE Mode IS SHIP

FastMilo ADDSLOTVALUE Cost IS MED_COST

FastMilo ADDSLOTVALUE OffensiveCapability IS MIN

FastMilo ADDSLOTVALUE DefensiveCapability IS MIN

FastMilo ADDSLOTVALUE Speed IS 938000

FastMilo ADDSLOTVALUE Range IS LONG_RANGE

FastMilo ADDSLOTVALUE Cargo IS LIGHT

All of the agencies that exist in this world must be defined as shown in the example below.

$ CREATEFRAME InterventionWing1 INSTANCE

InterventionWing1 ASSERTRELATION AKO Agency

InterventionWing1 ADDSLOTVALUE Name IS =1st=Intervention=Wing

InterventionWing1 ADDSLOTVALUE Commander IS Teri=Weeks

InterventionWing1 ADDSLOTVALUE FAD IS 0

InterventionWing1 ASSERTRELATION LOCATION Nellis

InterventionWing1 ASSERTRELATION OWNED BY StarForceInterventionCommand

The scenario database also contains specifications for initial stores of consumables and fleets of vehicles as shown below.

$ CREATEFRAME FuelCake1 INSTANCE

FuelCake1 ASSERTRELATION AKO Consumable

FuelCake1 ADDSLOTVALUE OwningAgency IS StarDepotSLC11

FuelCake1 ADDSLOTVALUE Kind IS FuelCakes

FuelCake1 ADDSLOTVALUE Location IS Provo

FuelCake1 ADDSLOTVALUE Number IS 3300

FuelCake1 ADDSLOTVALUE TimeCreated IS 0

FuelCake1 ADDSLOTVALUE Manufacturer IS EMPTY

FuelCake1 ADDSLOTVALUE ConditionCode IS 0

$ CREATEFRAME FastMilo_Luna INSTANCE

FastMilo_Luna ASSERTRELATION AKO Vehicle

FastMilo_Luna ADDSLOTVALUE Kind IS FastMilo

FastMilo_Luna ADDSLOTVALUE OwningAgency IS SpaceliftWing

FastMilo_Luna ADDSLOTVALUE Location IS Luna

FastMilo_Luna ADDSLOTVALUE Number IS 9

$ CREATEFRAME BusterBomb_Use1 INSTANCE

//BusterBomb_Use1 ASSERTRELATION AKO Consumption

//BusterBomb_Use1 ADDSLOTVALUE ProtoCommodity IS BusterBomb

//BusterBomb_Use1 ADDSLOTVALUE OwningAgency IS

BaseSupplyAgencySB10

//BusterBomb_Use1 ADDSLOTVALUE ConsumeAmount IS 12.5

//BusterBomb_Use1 ADDSLOTVALUE ConsumeInterval IS 1

//BusterBomb_Use1 ADDSLOTVALUE SupplyWindow IS 30

//BusterBomb_Use1 ADDSLOTVALUE PhaseShift IS 0 $ CREATEFRAME

Supply_BusterBomb INSTANCE

Supply_BusterBomb ASSERTRELATION AKO Supply

Supply_BusterBomb ADDSLOTVALUE FromAgency IS ACME

Supply_BusterBomb ADDSLOTVALUE ToAgency IS StarDepotSLC15

Supply_BusterBomb ADDSLOTVALUE CommodityType IS BusterBomb

Supply_BusterBomb ADDSLOTVALUE SupplyAmount IS 14

Supply_BusterBomb ADDSLOTVALUE SupplyInterval IS 24

Supply_BusterBomb ADDSLOTVALUE MaxAmount IS 0

Supply_BusterBomb ADDSLOTVALUE MaxInterval IS 0

Supply_BusterBomb ADDSLOTVALUE PhaseShift IS 0$ CREATEFRAME

Ship_BusterBomb2 INSTANCE

Ship_BusterBomb2 ASSERTRELATION AKO ShipmentSchedule

Ship_BusterBomb2 ADDSLOTVALUE AgencyList ALL-OF (StarDepotSLC15

Ogden Luna Herpes BaseSupplyAgencySB10)

Ship_BusterBomb2 ADDSLOTVALUE Source IS StarDepotSLC15

Ship_BusterBomb2 ADDSLOTVALUE Destination IS BaseSupplyAgencySB10

Ship_BusterBomb2 ADDSLOTVALUE MissionScheduleList ALL-OF

(RunRover6 FlyShorty3 Warp2 FlyShorty2)

Ship_BusterBomb2 ADDSLOTVALUE TheCommodity IS BusterBomb

Ship_BusterBomb2 ADDSLOTVALUE Frequency IS 12

Ship_BusterBomb2 ADDSLOTVALUE PhaseShift IS 2

Ship_BusterBomb2 ADDSLOTVALUE NumberResources IS 7

$ CREATEFRAME Warp2 INSTANCE

Warp2 ASSERTRELATION AKO FreightMissionSchedule

Warp2 ADDSLOTVALUE OwningAgency IS SpaceliftWing

Warp2 ADDSLOTVALUE FromAgency IS Luna

Warp2 ADDSLOTVALUE ToAgency IS Luna

Warp2 ADDSLOTVALUE TheProtoVehicle IS Milorex

Warp2 ADDSLOTVALUE RouteList ALL-OF (Luna_to_Herpes

Luna_to_Herpes)

Warp2 ADDSLOTVALUE Frequency IS 48

Warp2 ADDSLOTVALUE PhaseShift IS 0

3.5. Case Pool

The case pool is a part of the scenario database file that contains specifications for all of the objects that make up the cases. From this pool, cases may be chosen for presentation during the DMEs of an exercise. This file contains specifications for instances of each type of object described in Section 3.2. The syntax of these specifications is identical to that for the objects in the scenario. The following shows examples of case objects as they appear in the case pool.

$ CREATEFRAME Maldroit_resp2 INSTANCE

Maldroit_resp2 ASSERTRELATION AKO TextPacket

Maldroit_resp2 ADDSLOTVALUE PacType IS SITUATION

Maldroit_resp2 ADDSLOTVALUE PacName IS Maldroit_resp2

Maldroit_resp2 ADDSLOTVALUE PacCase IS Case2

Maldroit_resp2 ADDSLOTVALUE TextFile IS maldresp.txt

$ CREATEFRAME Case2InitialMessage INSTANCE

Case2InitialMessage ASSERTRELATION AKO Message

Case2InitialMessage ADDSLOTVALUE InfoPac IS Case2Msg1TP

Case2InitialMessage ADDSLOTVALUE ShowTime IS 0

Case2InitialMessage ADDSLOTVALUE Priority IS Low=Priority

Case2InitialMessage ADDSLOTVALUE ID IS AN=ID

Case2InitialMessage ADDSLOTVALUE Label IS Food=kit=contamination

Case2InitialMessage ADDSLOTVALUE TheAgency IS MaterielStorageSB5

$ CREATEFRAME Case2_Topic2 INSTANCE

Case2_Topic2 ASSERTRELATION AKO Topic

Case2_Topic2 ADDSLOTVALUE TopicName IS

MALDROIT=foodkit=contamination

Case2_Topic2 ADDSLOTVALUE ActivationConditions IS "Case2Msg1TP"

$ CREATEFRAME Case2_ID1 INSTANCE

Case2_ID1 ASSERTRELATION AKO InformationDeposit

Case2_ID1 ADDSLOTVALUE TheMessage IS Case2_ID1_FU_Msg

Case2_ID1 ADDSLOTVALUE TheTopic IS Case2_Topic1

Case2_ID1 ADDSLOTVALUE OpeningPacket IS Case2_ID1_IR

Case2_ID1 ADDSLOTVALUE RebuffPacket IS Case2_ID1_RB

Case2_ID1 ADDSLOTVALUE TheAgency IS MaterielStorageSB10

$ CREATEFRAME Case1 INSTANCE

Case1 ASSERTRELATION AKO DDTCase

Case1 ADDSLOTVALUE TopicList ALL-OF (Gossip_Topic1 Case1_Topic1 Case1_Topic2)

Case1 ADDSLOTVALUE MessageList ALL-OF (Case1InitialMessage Case3_Msg1)

Case1 ADDSLOTVALUE InfoDepositList ALL-OF (Case1_ID1 Case1_ID2 Case1_ID3 Case1_ID4 BattleNews21stID SB5GossipID SB6GossipID SB7GossipID SB8GossipID)

Case1 ADDSLOTVALUE SimChangerList ALL-OF (FCRemChanger)

$ CREATEFRAME FCRemChanger INSTANCE

FCRemChanger ASSERTRELATION AKO RemoveChanger

FCRemChanger ADDSLOTVALUE TheProtoResource IS FighterCans

FCRemChanger ADDSLOTVALUE Amount IS 2000

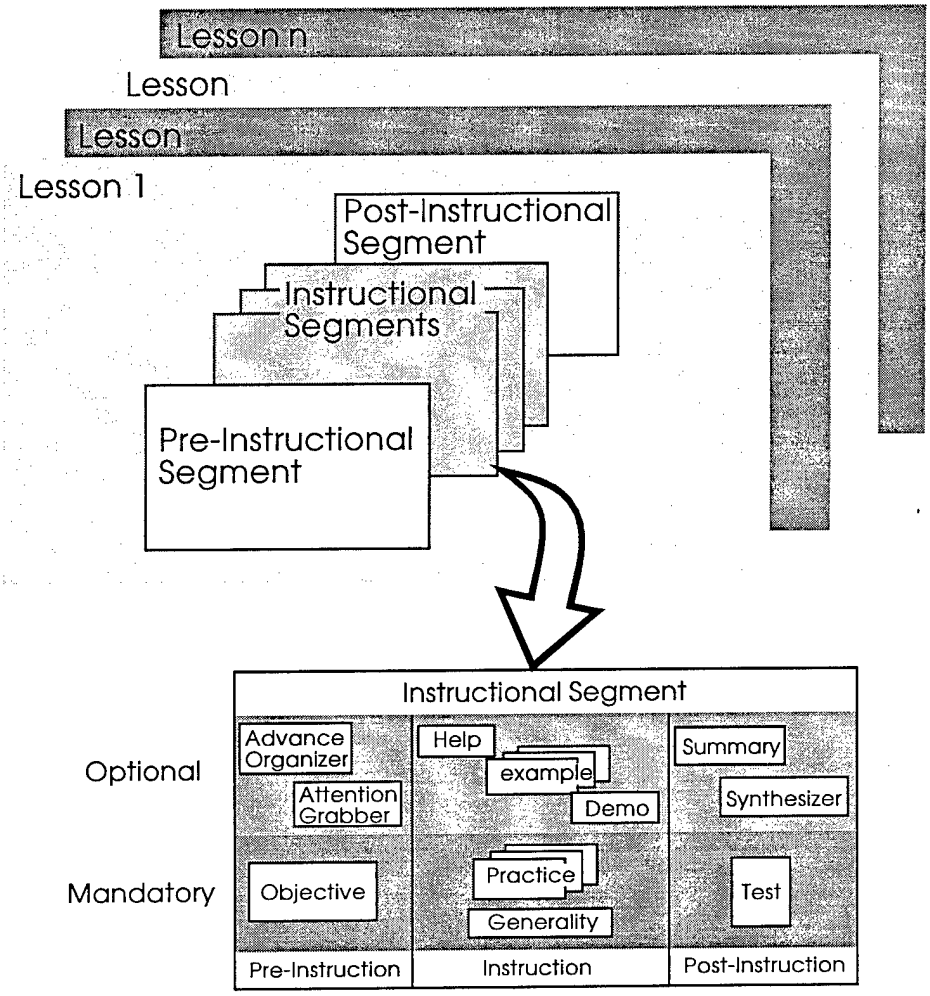
FCRemChanger ADDSLOTVALUE Location IS StarBase10

FCRemChanger ADDSLOTVALUE Owner IS BaseSupplyAgencySB10

4. Lesson Architecture

The DDT lessons are created and presented using ToolBook. ToolBook applications are subdivided into books, backgrounds, pages, and objects. An object is a graphical or interactive element on the screen such as a button or a text field. A page is a screen containing objects. A page sits on top of a background, which may also contain objects. Several pages may use the same background. Therefore, objects that appear on a large number of pages can be put on the background and need only be specified once. A book is a collection of pages that fits into a single file.

Figure 12. Lesson organization.



As figure 12 illustrates, the DDT lessons are subdivided into lessons, segments, and lesson elements. These subdivisions of a ToolBook application are used in a

structured fashion to create the subdivisions of a DDT lesson. Each lesson consists of one book. Each segment consists of a collection of backgrounds and pages. Each segment includes:

- A background and single page that contains all of the pre-instructional and post-instructional elements as objects.
- A background and collection of pages that contain the generality.
- A background and collection of pages that contain the practice items, where each practice item is a page.
- A background and collection of pages that contain the test items, where each test item is a page.

Where elements contain relatively large amounts of information, or where there are large numbers of very similar elements, they are implemented as a background and a collection of pages, while those elements that contain a small amount of information are implemented as objects on a single page.

The lessons also contain a large number of graphics, and a variety of commercial tools are used to create them. Ultimately all graphics are imported into ToolBook applications as Windows bitmap files, so Windows Paintbrush is almost always used for final composition and touchup. Corel Photo-Paint is used to edit photo-realistic images and to adjust the dithering effect necessary to display them on a 16-color device. Many of those images come from public-domain NASA archives, while others are created using tools such as Imagine and CorelDraw. CorelDraw is a sophisticated two-dimensional graphics design and illustration tool, and Imagine is a three-dimensional solids modeling and rendering tool.

References

Brecke, F. H., & Garcia, S. K. (1995). Training methodology for logistic decision making. (AL/HR-TR-1995-0095). Armstrong Laboratory, Human Resources Directorate, Brooks Air Force Base, Texas.

Brecke, F. H., & Garcia, S. K. (in press). Desktop decision training system requirements document. Armstrong Laboratory, Human Resources Directorate, Brooks Air Force Base, Texas.

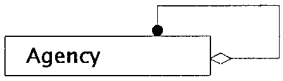
Appendix A. Detailed Object Model Descriptions

This appendix gives a more detailed description of the object model discussed in the body of this document. This includes descriptions of the objects and the events that cause them to change state.

A.1 Objects

For each object in the following sections there is a description, a list of attributes, and a state transition diagram. In some cases, the list of attributes or state transition diagram may be omitted where appropriate for the object.

Agency:



An agency is a place or an organization. An agency can be an operational unit such as a starfighter wing, a supply depot, a transportation command, or a supplier or manufacturer. It can also be an aggregate of several other agencies. In this case, the agencies in the aggregate are considered to be collocated, and there is no need to transport resources between them. Agencies have ownership/jurisdiction over resources and other agencies.

Attributes:

Name

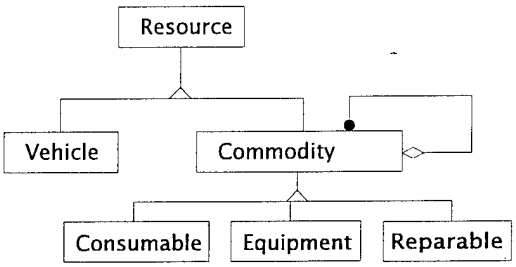
Commander

Commander Picture

Priority (Force Activity Designator (FAD))

Minimum Stopover Time

Resource:



A resource is a superclass that represents a physical entity, e.g., starfighters, fuel canisters, beer, compressed biomass cakes, fighter pilots, mechanics.

Attributes:

Number

Usable/Unusable Tag

Vehicle:

A vehicle is a resource that is capable of moving commodities from one agency to another.

Commodity:

A commodity is a resource whose supply the student must manage.

Attributes:

Creation Date

Condition Code

Manufacturer

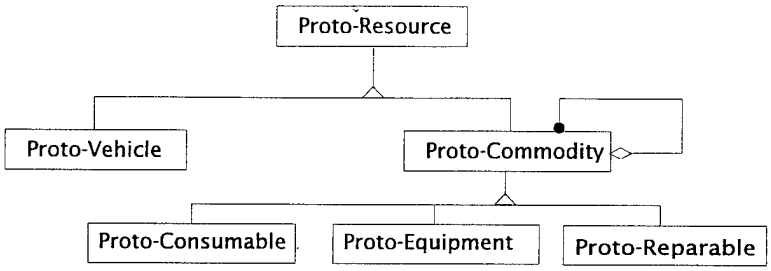
Consumable:

A consumable is a commodity that is used up, e.g., beer, fuel, food. For consumables there is a one-way transportation flow from supplier to consumer.

Repairable:

An repairable is a commodity that is refurbished after it has served its purpose, e.g., starfighter engines, guidance systems, R2D2 units. For repairables, there is a two-way transportation flow from supplier to consumer; refurbished units are sent to the consumer, "spent" units are returned to the supplier.

Proto-Resource:



A proto-resource is an object that describes or serves as a pattern for an actual resource. For example, there would be an instance of a proto-resource for each type of resource, such as fuel canisters, starfighters, hyper-warp freighters, or mechanics. A proto-resource contains the unchanging attributes that are common to all actual resources of a particular type. A proto-resource participates in associations that have the same semantics as the indefinite article "a," e.g., "I need a starfighter engine." Resources, on the other hand, participate in associations that have the same semantics as the definite article "the" or "that," e.g., "I am shipping that starfighter engine." As the diagram illustrates, the subclass tree for proto-resources is homomorphic with the subclass tree for resources.

Attributes:

Name

Proto-Vehicle:

Attributes:

Weight Capacity

Volume Capacity

Speed

Range

Cargo

Drive

Cost

Defensive Capability

Offensive Capability

Proto-Commodity:

Attributes:

Weight

Volume

Colloquial Name

Stock ID Number

Criticality

Proto-Consumable:

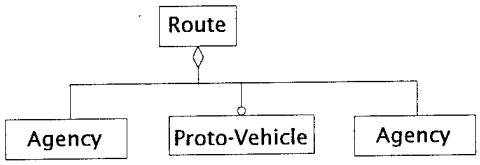
Attributes:

Shipping Container

Quantity per Container

Shelf Life

Route:



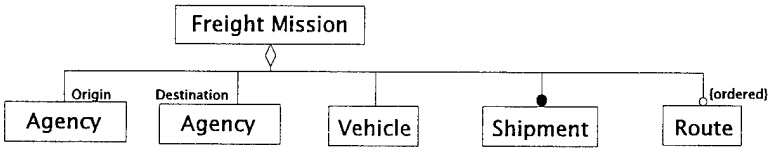
A route consists of two agencies and a list of one or more proto-vehicles and describes a transportation corridor between the agencies that supports the listed proto-vehicles.

Attributes:

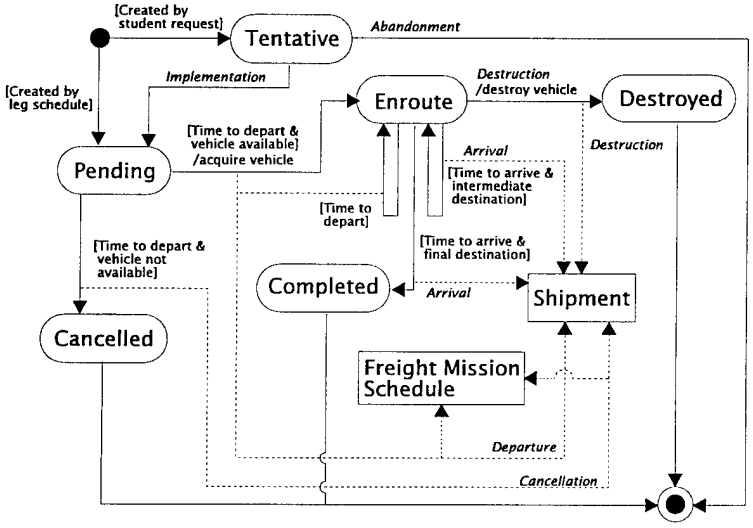
Distance

Safety

Freight Mission:

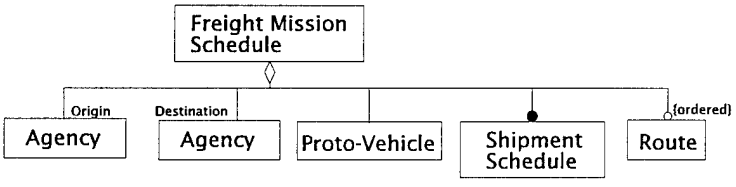


A freight mission consists of two agencies, an origin and a destination, a vehicle, and an ordered list of routes. A freight mission describes a trip from the origin agency to the destination agency following the routes on the ordered list.



State Transition Diagram

Freight Mission Schedule:



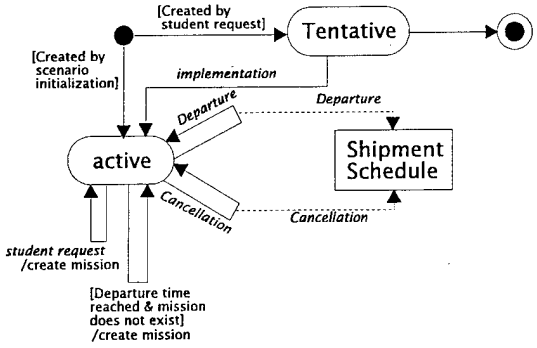
A freight mission schedule consists of two agencies (an origin and a destination), a proto-vehicle, and an ordered list of routes. A freight mission schedule may also be connected to zero or more shipment schedules. A freight mission schedule describes a

recurring freight mission. A freight mission schedule will "spawn" freight missions according to its schedule.

Attributes:

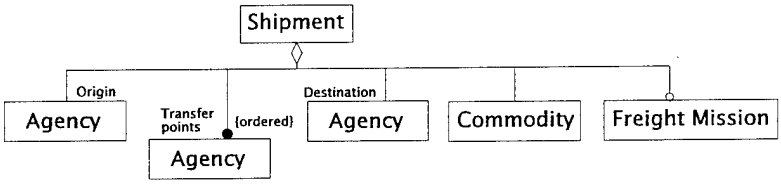
Frequency

Phase Shift



State Transition Diagram

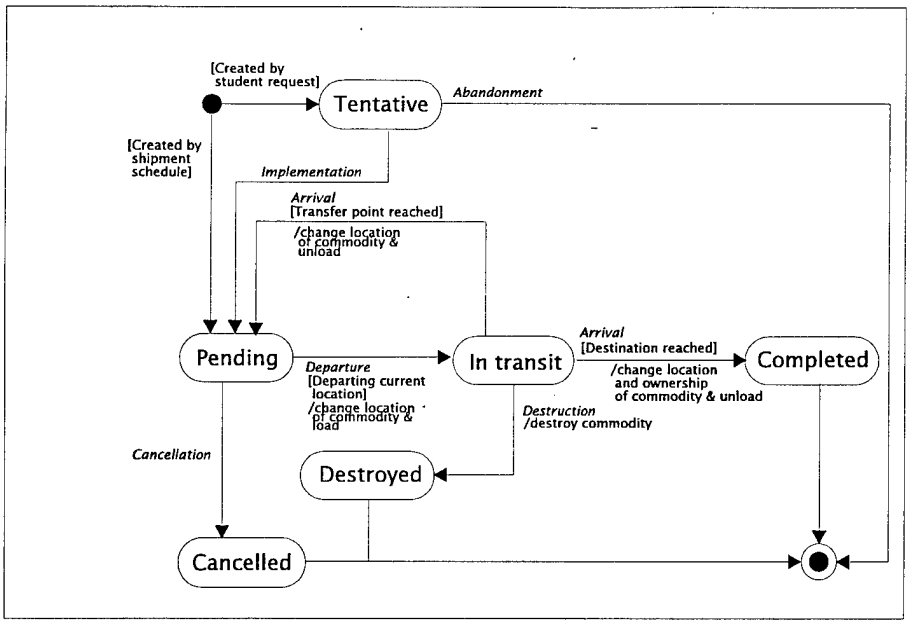
Shipment:



A shipment consists of two agencies (an origin and a destination), a commodity, one or more freight missions, and an ordered list of zero or more transfer point agencies. A shipment describes the movement of a commodity from the origin agency to the destination agency. The constituents of a shipment must adhere to certain constraints of space and time. Since a commodity cannot be in two places at once, all departure/arrival intervals of the freight missions at transfer points must be mutually exclusive. A commodity cannot magically jump from one place to another. Adjacent freight missions in the ordered list, therefore, intersect at a transfer point. A shipment transfers ownership of the commodity from the origin to the destination agency.

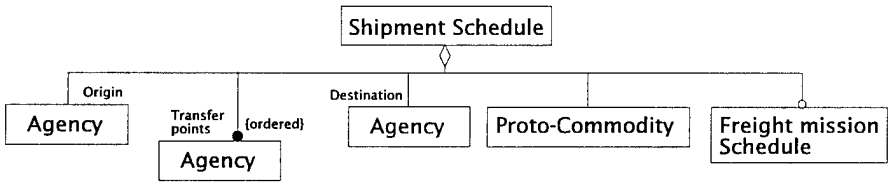
Attributes:

Urgency



State Transition Diagram

Shipment Schedule:



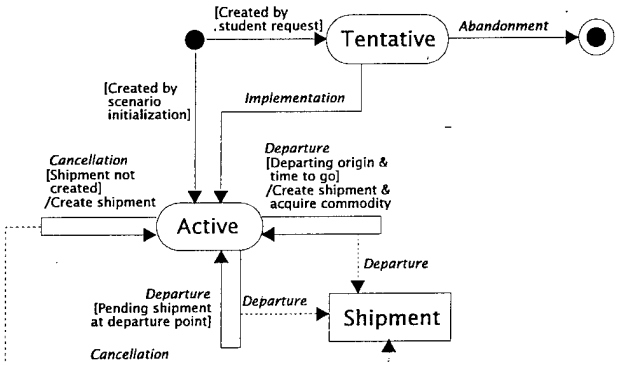
A shipment schedule describes or serves as a pattern for actual shipments. A shipment schedule describes the recurring movement of a commodity from one agency to another. A shipment schedule consists of two agencies, (an origin and a destination), a proto-commodity, one or more freight mission schedules, and an ordered list of transfer point agencies. The same time/space restrictions apply to the shipment schedule as apply to the shipment.

Attributes:

Frequency

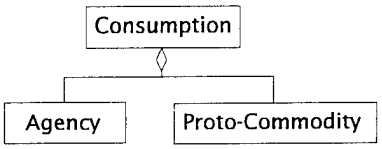
Phase Shift

Quantity

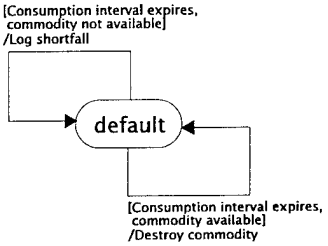


State Transition Diagram

Consumption:



A consumption object consists of an agency and a proto-commodity and describes the consumption of that type of commodity by the agency. A consumption association object is responsible for destroying commodity objects and serves as a sort of a drain of commodities out of the simulation.



Attributes:

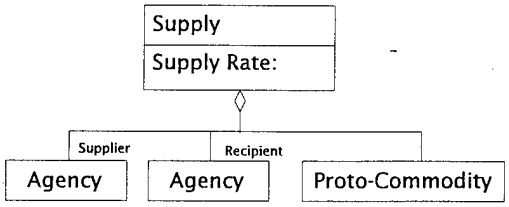
Consumption Amount

Consumption Interval

Phase Shift

Desired Stockpile Level

Supply:



A supply object consists of two agencies and a proto-commodity. One of the agencies is the supplier and the other is the recipient. The supply association describes the recurring flow of commodity objects from the supplier to the recipient. This transfer of commodities occurs by "magic" without the need for vehicles, freight missions, or shipments. A supply association is responsible for creating the commodity objects and serves as sort of a pump of commodities into the simulation.

Attributes:

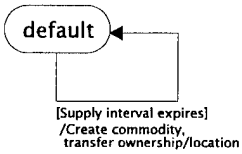
Supply Amount

Supply Interval

Phase Shift

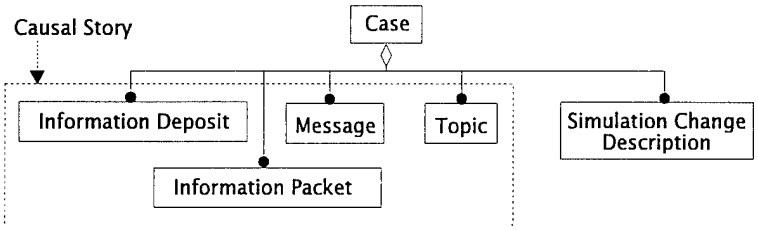
Maximum Supply Amount

Maximum Supply Interval



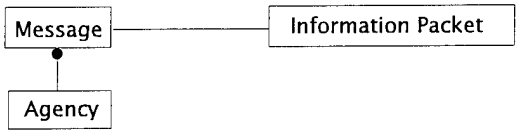
State Transition Diagram

Case Object:



The preceding diagram shows the objects that make up a case. The case object is responsible for managing active topics and for providing feedback about problem identification and uncertainty reduction.

Message:



A message is essentially a container for an information packet. A message has an attribute describing the subject of the message, an attribute that indicates when during the DME it should be delivered, and an association with the sending agency. Some messages in a case are simply delivered when their delivery time arrives, while others are associated with information deposits and are delivered only after the student has discovered the deposit. The next section, information deposit, describes this mechanism in more detail. Messages appear as closed envelopes on the message desk when they have been delivered.

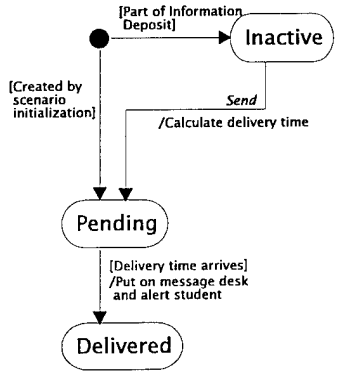
Attributes:

Priority

ID Number

Subject

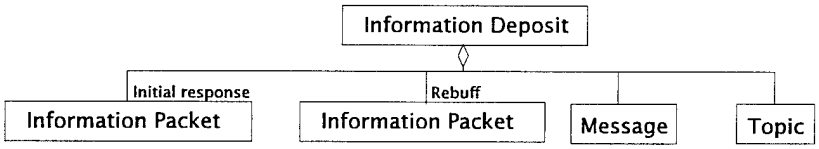
Delivery Time



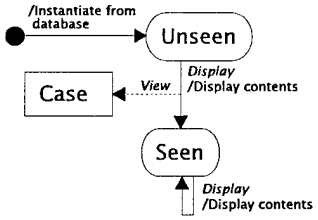
State Transition Diagram

Messages are instantiated from the database when a case is read in. Some messages are instantiated as part of an information deposit and start in the inactive state, while others are instantiated in the pending state. Messages in the pending state do nothing until they receive a "send" event with a delivery time from their information deposit. Then they are associated with their sending agency, go into the pending state, and await delivery. Messages in the pending state monitor simulation time clicks until their delivery time arrives, then notify the message desk and transition to the delivered state.

Information Deposit:



An information deposit consists of an eliciting topic, an immediate response, a follow-up message, and a rebuff. The eliciting topic is the topic for which the information deposit is a response. The immediate response is an information packet that is presented to the student as soon as the query is made. The follow-up message is a message that is "sent" to the student at some time after the initial query. The rebuff is an information packet that is presented to the student if the student asks about the same topic again.



State Transition Diagram

Each information deposit is instantiated from the database and starts in the undiscovered state. When an information deposit receives a discovery event from its agency, it checks the topic to see if it matches its eliciting topic. If it matches, and the information deposit is in the undiscovered state, the information deposit transitions to the

discovered state, sending a "send" event to its follow-up message and a display event to its immediate response information packet. When an information deposit receives a discovery event in the discovered state and the eliciting topic matches, it sends a display event to its rebuff information packet and remains in the discovered state.

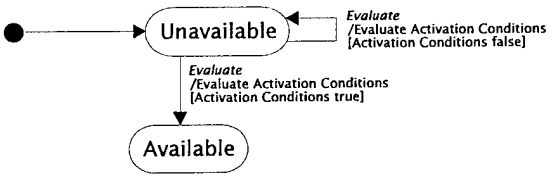
Topic

A topic is a word or short phrase that can serve as the object of a student's query of an agency. The topic contains a description, called the activation conditions, of the conditions under which it becomes available to the student as a possible object of a query. These conditions are represented as a Boolean expression of information packet identifiers and the operators "and" and "or." The information packet identifiers "evaluate true" if the information packet has already been viewed by the student. For example, the topic of "destroyed fuel canisters" would become available after the student had viewed a message about an enemy attack at Starbase 10 or after the student had read a report listing the shortage.

Attributes:

Key word

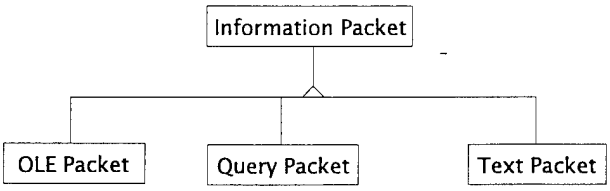
Activation conditions



State Transition Diagram

A topic is in the available state when its activation conditions evaluate true and it is in the unavailable state when they evaluate false. The case object is responsible for maintaining state information for all the topics in the case. When a topic receives a evaluate event from the case, it evaluates its activation conditions and notifies the case object about the result. The following diagram illustrates this behavior.

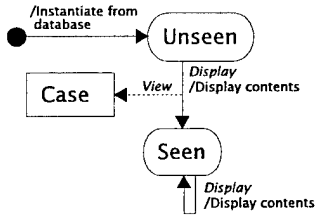
Information Packet



An information packet is actually a superclass, with a virtual display method. Subclasses of information packets are differentiated based on the form of the information they contain. Currently three subclasses are envisioned: an OLE packet, a query packet, and a text packet. An information packet is tagged with information about the types of information it contains (e.g., situation, option set, option feasibility, option effects, distracter), allowing the case to provide feedback about the student's uncertainty reduction.

Attributes:

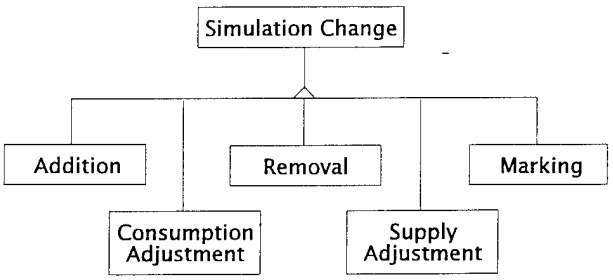
Information type



State Transition Diagram

An information packet is instantiated at the time the case is read in from the database. All information packets are specified in the database. When the packet receives a display event, it displays or otherwise presents its contents to the student. Additionally, if it is the first display event, it sends a "view" event to its case so that the case can update the active topics.

Simulation Change Objects:



These objects describe and effect the changes to the logistics simulation necessary to "cause" the logistics event associated with case. There are five different simulation change subclasses, each corresponding to a different type of change.

A.2 Events

In the preceding sections we described the possible patterns of objects and attributes that can exist in the system. A particular set of attribute values held by an object is its state. Over time the objects interact with one another, resulting in changes to their states. These interactions are called events. Usually, an event is generated by one object and stimulates a change in state of other objects that recognize that event. The following sections describe most of the events listed in the state transition diagrams in the preceding sections.

Departure (freight mission, agency)

This event is generated by a freight mission whenever the time arrives for it to depart an agency. For the origin agency, the departure time is an attribute of the freight mission. For intermediate agencies, the departure time is the arrival time plus any stopover or loading overhead. This event is sent to any shipments connected with the freight mission and to the freight mission's parent freight mission schedule if there is one. Shipments and shipment schedules are the final recipients of departure events (the freight mission schedule forwards the event to its shipment schedules). When a shipment receives a departure event, it decides whether or not to load itself onto the freight mission. When a shipment schedule receives a departure event, it decides whether or not

to create a shipment and pass it the departure event, to pass the departure event to an already created shipment, or to do nothing.

Arrival (freight mission, agency)

This event is generated by a freight mission whenever the time arrives for it to arrive at an agency. The arrival time is the departure time plus travel time. This event is sent to any shipments connected with the freight mission. When a shipment receives an arrival event, it decides whether or not to unload itself from the freight mission.

Cancellation (freight mission)

This event is generated by a freight mission when its departure time has arrived and there is no vehicle of the proper type that is owned by its parent agency, located at the origin of the freight mission, and not already committed to something else. This event is sent to any shipments connected with the freight mission and to the freight mission's parent freight mission schedule if there is one. When a shipment schedule receives a cancellation event, it creates a new shipment without acquiring the associated commodity and forwards the cancellation event to it. When a shipment receives a cancellation event, it sends a stockpile change event to the destination agency.

Destruction (freight mission):

This event is generated by a freight mission when it receives a scripted event indicating that it has been destroyed. This event is sent to any shipments connected with the freight mission and to the freight mission's parent freight mission schedule if there is one. When a shipment schedule receives a destruction event, it creates a new shipment without acquiring the associated commodity and forwards the destruction event to it. When a shipment receives a cancellation event, it destroys any associated commodity and sends a stockpile change event to the destination agency.

Implementation:

This event is sent to freight missions, freight mission schedules, shipments, and shipment schedules when they are in the tentative state and the student has decided to

implement them. This event causes the corresponding object to change to an active or a pending state.

Consume (proto-commodity, amount):

This event is generated by a consumption object and is sent to an agency to instruct it to destroy a certain amount of a particular commodity.